The Boomerang Project
A brief description of both the Boomerang Project and the final project is included in this module.

The Project:

The Boomerang Project is a program used to detect the position of a sniper based on the sound a sniper rifle makes when fired. Originally developed by BBN Technologies and now currently used on military vehicles, the Boomerang Project is a wonderful example of how filtering can be combined with signal processing in real time to produce real results. Pinpointing the origin of enemy fire is not a new idea, however, it is only in recent times that improved microphones, filtering, and data processing have allowed DARPA to develop accurate and quick technologies. Current setups are used on military vehicles, but a smaller setup is in development for individual soldiers.

In our project, we sought to recreate the Boomerang Project's algorithms in order to determine the direction of a handclap. We used an array of four microphones and the audiorecorder function in Matlab to determine the person's position in a 2D plane using a technique called multilateration. To increase the practicality of the project, we tried to make the program run as close to real time as possible, since when a sniper is shooting at you, sooner is better than later.

Behind the Scenes: Algorithms at Work
In this module is a description of multilateration, the technique used to pinpoint position.

Behind the scenes Algorithms:

Because we know only the time difference between each microphone's recognition of the sound, not the initial time of the sound itself, we decided to use multilateration as the primary algorithm. Multilateration uses the time difference of arrival to calculate a distance from the receivers. Plotting with a radius of this distance yields a hyperbolic arc that gives possible locations for the sound. The intersection of multiple arcs found by using multiple sensors gives the position of the origin of the sound.

In more detail, the microphones pick up the sound at different times which Matlab then cross correlates to get the time difference of arrival between any two sensors. This time difference of arrival is then used with the speed of the sound to create a series of linear equations that when solved, will give the direction and position of the sound's origin.

With a sound emitter (Es) at some unknown position (x,y), we create a coordinate system around the known positions of the sensors, with the position of one sensor defined as the origin (S0). Using this sensor as the origin simplifies the distance from this sensor to Es. The distance from the sensor at the origin to the emitter can be written as

d0=S0 - Es= x2+ y2

The distance d0 is the speed of sound multiplied by the transit time (T0). However, since we do not know T0, we use the time difference of arrival between S0 and any other sensor, Sm.

vτm=vTm- vT0

vτm= dm- d0

Next, we used the cross correlation function to measure the time shift between the recorded waveforms, which is τm. Some manipulation of the

above equations gives a set of linear equations of Es.

$$0 = xAm + yBm + C\ m$$

where Am, Bm, and Cm are constants representing the distances from the chosen sensor Sm and Es. Because S0 is used in each equation, S0 cannot be used to generate an equation for the set.

As an algorithm, multilateration requires very little input: the times at which each microphone recognizes the sound, the fixed distance between the microphones, and the speed of sound. As a result, it was easier to implement multilateration over more conventional triangulation methods in which the angle and the distances between the microphones and the origin of sound are needed.

Program
This module gives a summary of the code we wrote to implement multilateration.

Program:

Here is the code for the program that we wrote to record and analyze the data received from the microphones:

%%%

% Program constants

%%%

nMics = 4; % The number of microphones

Fs = 44100; % The sample rate

nBits = 8; % The number of bits per sample, either {8,16,24}

nChannels = 1; % The number of channels recorded, either {0,1}

recordLengthSeconds = 3; % The length of each recording in seconds

recordLength = recordLengthSeconds*Fs; % The length of each recording in entries

v = 1130.90551;

maxLags = 0.5*Fs;

% Offset in seconds to add to each mic's tau to account for delay start in recording

mic1TauOffset = 0;

mic2TauOffset = 0.0279;

mic3TauOffset = 0.0596;

mic4TauOffset = 0.1095;

This first part demonstrates the variables that we created and our averages (the offsets) used in calibrating the microphones (we got these values with time delay tests before we tested the entire array). The variable v represents the speed of sound in the area that we tested, and the maxLags represents the maximum lag between the microphones.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Initialization

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

pause on; % Enable pausing, which is used below

ids = zeros(1,nMics); % The device ID for each microphone

ids(1) = 1;

ids(2) = 2;

ids(3) = 3;

ids(4) = 4;

taus = zeros(1,nMics); % The TDOA in seconds between each mic and mic 1

```
As = zeros(1,nMics-2);

Bs = zeros(1,nMics-2);

Ds = zeros(1,nMics-2);

xs = zeros(1,nMics); % The x-position in feet of each mic relative to mic 1

ys = zeros(1,nMics); % The y-position in feet of each mic relative to mic 1

%{

xs(1) = 0; % Microphone #1 is at (0,0)

ys(1) = 0;

xs(2) = 0; % Microphone #2 is at (0,2)

ys(2) = 2;

xs(3) = 2; % Microphone #3 is at (2,0)

ys(3) = 0;

xs(4) = 2; % Microphone #4 is at (2,2)

ys(4) = 2;

%}

xs(1) = 0; % Microphone #1 is at (0,0)

ys(1) = 0;

xs(2) = -.5; % Microphone #2 is at (-.5,1)

ys(2) = 1;

xs(3) = -1; % Microphone #3 is at (-1,2)
```

ys(3) = 2;

xs(4) = -1; % Microphone #4 is at (-1,3)

ys(4) = 3;

A = zeros(nMics-2,2); %

B = zeros(nMics-2,1); % The matrices used to solve for point of origin, AX = B

This second part shows how we set up our coordinate grid. We used Matlab to identify each microphone and them input the position of each microphone (both array setups are shown). The last three lines identify the matrices used to find the point of origin, meaning that position was solved with respect to a certain microphone (the microphone at (0,0)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Record audio data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

recorder1 = audiorecorder(Fs, nBits, nChannels, ids(1));

recorder2 = audiorecorder(Fs, nBits, nChannels, ids(2));

recorder3 = audiorecorder(Fs, nBits, nChannels, ids(3));

recorder4 = audiorecorder(Fs, nBits, nChannels, ids(4));

record(recorder1, recordLengthSeconds);

record(recorder2, recordLengthSeconds);

```
record(recorder3, recordLengthSeconds);
record(recorder4, recordLengthSeconds);
pause(recordLengthSeconds*1.5); % Pause for the length of the recording
% plus 50% to ensure recording finishes
temp = getaudiodata(recorder1);
record1 = temp;
temp = getaudiodata(recorder2);
record2 = temp;
temp = getaudiodata(recorder3);
record3 = temp;
temp = getaudiodata(recorder4);
record4 = temp;
%{
figure;
plot(record1);
figure;
plot(record2);
figure;
plot(record3);
figure;
```

plot(record4);

%}

The third part involves the recording the data from the microphones and then plotting the impulse. The pause function is used to ensure that all of the impulse is recorded and not cut off. The data is then plotted on the same graph. The plots all showed the impulse response of the clap.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Analyze audio data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

% Find TDOA for each mic

taus(1) = 0; % No TDOA between mic 1 and itself

crosscorrelation = xcorr(record2,record1,maxLags); % TDOA for mic 2

[maxVal, maxIndex] = max(crosscorrelation);

taus(2) = (maxIndex-maxLags)/Fs + mic2TauOffset;

crosscorrelation = xcorr(record3,record1,maxLags); % TDOA for mic 3

[maxVal, maxIndex] = max(crosscorrelation);

taus(3) = (maxIndex-maxLags)/Fs + mic3TauOffset;

crosscorrelation = xcorr(record4,record1,maxLags); % TDOA for mic 4

[maxVal, maxIndex] = max(crosscorrelation);
```

```
taus(4) = (maxIndex-maxLags)/Fs + mic4TauOffset;

% Find A_m, B_m and D_m for each mic m

for i=3:nMics

As(i) = 2*xs(i)/(v*taus(i)) - 2*xs(2)/(v*taus(2));

Bs(i) = 2*ys(i)/(v*taus(i)) - 2*ys(2)/(v*taus(2));

Ds(i) = v*taus(i) - v*taus(2) - (xs(i)^2 + ys(i)^2)/(v*taus(i)) + (xs(2)^2 +
ys(2)^2)/(v*taus(2));

end

% Generate matrices A, B to solve for Ax = B

for i=3:nMics

A(i-2,1) = As(i);

A(i-2,2) = Bs(i);

B(i-2,1) = -Ds(i);

end

X = linsolve(A,B);

A

B

X

taus
```

The final part involved finding the actual time differences between the sound recordings of the microphones. As seen at the top, the known data of

the different time arrivals of the sound to each individual microphone was cross-correlated in order to find the time differences in arrival of the sound for any two microphones. This information was then used to create matrix points for each of the microphones. The matrix was then solved to find the position of the person clapping using the equation $Ax = B$. The values were then printed out.

Testing and Results
This module describes our testing setup and the results we obtained.

Testing and Results:

Our testing array consisted of four microphones borrowed from the electrical engineering lab, four audio to USB converters, and a USB hub that was then connected to the computer running the program. We tested two different setups (positions) for the array of microphones, a square and a parallelogram. Again, we simulated the rifle impulse with a hand clap. Our results are as follows:

First Array: Square

Microphones at (0, 0, 0), (0, 2, 0), (2, 0, 0), and (2, 2, 0)

Position	**(X,Y,Z) Results**	**Tau (1,2,3,4)**
(0,0,0)	(-651.2859, -313.7830, 0)	0 0.0118 0.0103 -0.2602
(0,0,0)	(-199.8994, 272.4750, 0)	0 0.0064 -0.0028 0.4928
(0.5,-7,0)	(1.0e+003*-1.1840, 1.0e+003*0.0451, 0)	0 0.0030 -0.0117 0.4828
(0.5,-7,0)	(1.0e+003*3.8188, 1.0e+003*1.3335, 0)	0 0.0218 0.0303 0.4928
(8,1,0)	(1.0e+004*1.1917, 1.0e+004*0.0959, 0)	0 0.0018 0.1032 0.4828

(8,1,0)	(1.0e+003*7.0765, 1.0e+003*0.0780, 0)	0 0.0118 -0.0726 -0.4460
(1.5,19,0)	(1.0e+004*-1.0404, 1.0e+004*-0.1080, 0)	0 0.0118 0.1132 -0.4460
(1.5,19,0)	(1.0e+003*2.4049, 1.0e+003* 0.7235, 0)	0 0.0118 0.0203 0.4828
(-3,2,0)	(1.0e+003*3.7463, 1.0e+003* 1.3085, 0)	0 0.0218 0.0303 0.4828
(-3,2,0)	(1.0e+004*-1.8326, 1.0e+004*0.1534, 0)	0 0.0118 0.2061 -0.4460
(-4,-2,0)	(1.0e+003*-1.0406, 1.0e+003*-0.1446, 0)	0 0.0018 0.0103 -0.4560
(-4,-2,0)	(1.0e+003*3.5954, 1.0e+003*0.8343, 0)	0 0.0118 0.0303 0.4928

Second Array: Parallelogram

Microphones at (0, 0, 0), (-0.5, 7, 0), (3, -1, 0), (3.5, 4, 0)

Position	**(X,Y,Z) Results**	**Tau (1,2,3,4)**
(0,0,0)	(-201.4001, 63.0702, 0)	0 -0.0911 0.0096 0.0068
(0,0,0)	(2.7784, -7.2170, 0)	0 -0.0070 0.0081 -0.0038

(0.5,-7,0)	(1.0e+003* 0.7510, 1.0e+003*-3.3145, 0)	0 -0.0711 0.0203 0.4828
(0.5,-7,0)	(1.0e+004*1.5821, 1.0e+004*-0.2474, 0)	0 0.0218 0.1232 0.4828
(8,1,0)	(-50.6356, -11.9636, 0)	0 0.0018 -0.0726 -0.0745
(8,1,0)	(858.3622, 88.1823, 0)	0 0.0118 -0.0726 -0.0745
(1.5,19,0)	(-40.0023, -925.3581, 0)	0 -0.0711 0.0303 0.1213
(1.5,19,0)	(1.0e+003*2.6500, 1.0e+003*1.2213, 0)	0 0.0218 0.0203 0.4928
(-3,2,0)	(1.0e+003*2.4049, 1.0e+003*0.7235, 0)	0 0.0118 0.0203 0.4828
(-3,2,0)	(1.0e+004*-1.2287, 1.0e+004*0.1966, 0)	0 0.0218 0.1232 0.3071
(-4,-2,0)	(1.0e+003*-5.6438, 1.0e+003*-0.0804, 0)	0 0.0118 -0.0726 -0.3531
(-4,-2,0)	(1.0e+003*1.7925, 1.0e+003*0.2009, 0)	0 0.0018 0.0203 0.3899

As can be seen above in both of our test setups the scale (originally measured in feet) was off by a large magnitude. The second setup, the parallelogram did resulted in less of an error in scale, but there was still a large discrepancy. Although the scale was off, the ratio of X and Y (shown by the position of the clap) was correct even if the magnitude was off.

Besides discrepancies in scale we did have issues with our results for tau, which represented the time delay of the audio reaching the various microphones. We were unable to synchronize the microphones and get them all to begin recording simultaneously. We did try calibrating them but the recording delays of the microphones changed each time so our calibration was an average, and therefore not exact. This meant that the microphones had a delay (of about a tenth of a second) between the first one beginning to record to the last one. In some cases, the microphone that should have recorded the sound first began recording before the other microphones, making it appear that that microphone heard the sound last. These issues overall led to skewed results for the various values of tau in our testing. Because the results depended so much on the time delay, none of our results gave the correct location of the origin of the sound. We did however manage to get the general direction of the sound correct about 50% of the time.

The results that we produced were far from what we expected or desired. Although we did manage to get the general direction in about half the cases we were nowhere near determining the exact position. The problem was most likely caused by the inability to the properly calibrate or synchronize the recording of the microphones. There are a variety of ways that could have solved this problem such as an external trigger that started the microphones at the same time or running the microphones on four different programs that fed into one other program that then analyzed the data. Microphone sensitivity could also possibly improve the results.

Overall, our results do determine the general direction of the sound and prove that multilateration can be used to help pinpoint location.

Complications
This module contains a summary of the complications that arose during our testing.

Complications:

As is inevitable in any project, we ran into a few complications along the way. It was not possible for us to test our program with a sniper shot, or mounted on a military vehicle. Instead, we used a handclap, which we believe provides the necessary difference in amplitude to be recognized against the noise inherent to the system. In the program itself, we had difficulties calibrating the microphones and setting them to record simultaneously. With different start recording times, it was difficult to determine a correct time difference of arrival, τ_m.

Another error arose when we tried to calculate the position. The time differences of arrival were slightly inaccurate due to the time differences that the computer creates because it cannot start all four microphones recording at the same time. Originally we set the microphones into a square shape, approximately 2 feet apart, but when this gave positions with more error than we felt was acceptable, we rearranged the microphones into a parallelogram shape with much larger distances, approximately 7 feet. Overall, since we were able to find correct ratios but not magnitudes, we feel that our testing setup needed to be on a larger scale than we were able to test with: if we had access to better microphones and been able to test with distances of 100 feet, instead of 10, we might have been able to troubleshoot errors better.

If we had the time to repeat our experiments, we would like to look into the possibility of using a different coding language, that would allow us to start all four microphones recording simultaneously, or at four different programs to start each microphone and then feed the data from those four subprograms into a driver program to do the multilateration calculations.

Implications
To conclude our project, we will briefly discuss the implications of our work to real world applications.

Implications

We feel that although our project did not give the results we desired, we showed the potential of multilateration as an algorithm working in real time to determine the position of a sound’s origin. This system has already been proved useful in combat situations, but we believe it could also be expanded to assist in search and rescue missions, theft recovery, and many more. The flexibility in the frequencies of the sounds the system can pinpoint allows the system to adapt to a variety of constraints, even allowing the frequencies to be above what the human ear can detect, giving the system an edge of stealth that could be useful in theft recovery and in kidnappings in which the police force wishes to capture the kidnapper as well as rescue the kidnapped. The myriad of uses for this kind of position locating have not been fully explored, and we look forward to seeing where the field goes in the future.